Arpanet and Its Evolution — A Report Card

Stephen D. Crocker

The author provides a "back to the future" report card on the Internet's seedling, the Arpanet. The vision and goals for the Arpanet are evaluated along several aspects of functionality, performance, and evolution. The resulting grades suggest what direction future evolution might take to allow the Internet to evolve into what it was intended to be.

Abstract

This article is a "back to the future" report card on the Internet's seedling, the Arpanet. The vision and goals for the Arpanet are evaluated along several aspects of functionality, performance, and evolution. The resulting grades suggest what direction future evolution might take to allow the Internet to evolve into what it was intended to be.

Introduction

This issue is about predicting the future of the Internet. I'm going to cheat and talk about predictions of the future from the perspective of 50 years ago when we were envisioning the network. It's a cheat, of course, because we now know how it turned out. At least we know how the first 50 years turned out. This perspective is interesting nonetheless because we can gauge what happened against what we intended and expected. And let me limit your expectations. This view is primarily about the technical aspects of networking. The current concerns about social networking, privacy invasions, spam, fraud, and so on appeared quite early but were minor annoyances compared to today's environment. Those of us involved in creating the Arpanet and then the Internet were almost totally focused on creating the capability and getting it to work.

To make this discussion fun and perhaps controversial, I offer a report card along several aspects of functionality, performance, and evolution. But first, let's go back in time to the creation of the Arpanet.

The Arpanet was DARPA's ambitious project to connect its computer science research laboratories with a general-purpose, highly interactive network. These few words — computer science, research laboratories, general-purpose, and highly interactive — may seem bland but carry a lot of content.

In its first few years, DARPA's focus was on space technology and other military technologies. In 1962 the Information Processing Techniques Office (IPTO) was created, with J.C.R. Licklider ("Lick") as the founding director [1].

Although DARPA spent most of its budget on fast-paced research aimed at significant results within three to seven years, a small portion of its budget was devoted to long-term investment in two basic technologies — material science and computer science — that would pay off over time. Its investments in computer science were in the form of funding several research laboratories — MIT, CMU, the University of California (UC) Berkeley, University of Utah, SRI, UC Los Angeles (UCLA), and others. The institutional support was important but not sufficient. There was also a vision....

The Vision

In the early days of computing, from the late 1940s through the 1950s and early 1960s, computers were expensive and available only to large organizations — governments, businesses, and universities. And it was important to keep them busy doing useful work. The idea of a personal computer was nothing but a dream, closer to science fiction than a realistic goal.

Commercially, IBM was the dominant computer company. Several other computer companies — Burrough's, Control Data Corporation, NCR, Univac, and others — competed as best they could. Eventually, they all faded.

These computers were all physically large and usually required air conditioning. The large ones lived in specially constructed rooms with false floors that provided airflow and had room for thick cables.

Keeping them busy meant organizing the workload in a fashion that staged a queue of jobs for the computer to carry out. The term of art was "batch processing." Programmers prepared programs by sitting at their desks writing code and then copying their programs onto punched cards or paper tape. When the computer was ready to execute their programs, it read in the program and then took however long was necessary to compute the results. The output was often printed on wide paper and then placed in folders or cubbyholes for the programmer to retrieve.

Efficiency was paramount. In the earliest days, programs were written directly in machine instructions. When high-order languages were introduced, it was understood these new languages would make it easier to write programs. Still, there was concern about whether the code generated by compilers would be competitive with hand-written code. The success of Fortran was due in part to the compiler generating code that met that bar.

The astute reader will have noticed I have not used the terms "interactive" or "interpretative." Most managers of computers considered it wasteful for the computer to wait for a human to decide what to do or for the computer to translate source code to object code interspersed with doing "real" work.

There was, however, a small portion of the computing community who shared a different vision. Programming was tedious and prone to errors. In a batch processing environment, the programmer might have to wait several hours or even a full day to get results back, only to discover there was an error in the program. Suppose — just imagine! — being able to see syntax errors immediately.

Well, as we know, computers have become drastically smaller, cheaper, and more available.

Digital Object Identifier:
10.1109/MCOM.001.2100727

Stephen D. Crocker is with Shinkuro Inc.

0163-6804/21/$25.00 © 2021 IEEE

Everyone reading this article undoubtedly has one or more personal computers, some of which are called "phones." But in the mid-1960s, the idea of a computer waiting for the human instead of the other way around was part of a vision understood and pursued in just a few pockets of academia.

The vision was even more ambitious than just having a computer at the ready; it included having the computer carry some of the mental labor. At the extreme end, the catchy phrase "artificial intelligence" (AI) conveyed the idea of computers thinking, however ill-defined that term might be. The early work on AI focused on programs that played checkers and chess, solved symbolic algebra and calculus problems, attempted to make rudimentary sense of written text, and attempted to infer the difference between objects and background in photos.

Toward the more practical end of the spectrum, time-sharing systems were developed and fielded. Similarly, interactive, interpretive languages such as LISP and BASIC were developed and fielded [2, 3]. In the mid-1960s, these technologies were in regular use in a few select places. The difference between those environments and the common environments of batch processing and efficient use of large computers was like night and day. The Arpanet was part of the vision of a collaborative environment that supported interaction between people and computers, between computers, and, not insignificantly, between people.

The Report Card

This report card has three sections. I assign grades in three broad areas: functionality, performance, and evolution. Each of these has multiple aspects.

Functionality

The four aspects are connectivity, protocols, collaboration, and extensibility.

Connectivity: Highly interactive computing was just the beginning of the vision. Computer-to-computer communication, online libraries, and collaboration across research laboratories were also part of the vision. How to do all of that was not as clear. Some hurdles were obvious; additional hurdles emerged later.

There had been a handful of prior experiments to connect computers together. Most were unsuccessful, and the others were of limited use for a short duration. The one that turned out to be the most useful was a connection between Lincoln Laboratory in Massachusetts and System Development Corporation in California. In addition to solving the problems of getting two dissimilar computers to interact, the work also provided useful data on the error rates of long-distance telephone lines.

Finally, in 1965, IPTO decided it was time for a large-scale effort to connect the computers across the participating laboratories. The short version of the decision process, recounted in Hafner's and Lyons' book *Where Wizards Stay Up Late*, is that Bob Taylor, the head of IPTO, explained the concept to his boss, Charles Herzfeld, the director of DARPA [4]. Herzfeld approved $1,000,000 on the project. The actual process was more detailed and extended, and Larry Roberts was recruited from Lincoln Laboratory to head up the project.

ARPA, DARPA — The Defense Advanced Research Projects Agency, was started in 1958 in response to the Russian launch of the first satellite, Sputnik. When it was started, it was inside the Office of the Secretary of Defense (OSD) and was called the Advanced Research Projects Agency. In 1972 it was moved out of OSD to become a separate Defense agency and its name was changed. This was an administrative change with no change in mission or structure. The name changed back briefly to ARPA in the early 1990s and then back again a short time later to DARPA. The Arpanet was created when the agency was ARPA, and in most writings the name of the network remained the same.
FTP (File Transfer Protocol) was one of the earliest defined protocols. The original specification was written in 1971 by Abhay and ran on NCP. The protocol was later replaced by a TCP/IP version; while still built into several operating systems, it is increasingly deprecated across the Internet.
IMP, router — The Arpanet introduced the concept of using a separate computer as a router. The term "router" was not yet in use. An Arpanet router was called an IMP, Interface Message Processor.
IPTO (Information Processing Techniques Office) was the office within the Advanced Research Projects Agency that funded advanced computer science research in the 1960s and 1970s.
Multics (Multiplexed Information and Computing Service) was an influential time-sharing system that incorporated advanced memory management and security controls.
NCP — Originally, the abbreviation NCP stood for Network Control Program and referred to the software that had to be added to the operating system of the host to communicate with the IMP and implement the host-host protocol between the hosts. Over time, the term NCP became repurposed to stand for Network Control Protocol to refer to the protocol and not the software.
NWG (Network Working Group) — Originally, this designated a small group of principal investigators who advised IPTO management during the 1965-68 design phase of the Arpanet design. The same name was then used by the less senior researchers, primarily graduate students, who developed the original suite of host-level protocols. The NWG evolved over time from around a dozen people in 1968 to a larger and larger group and eventually became the Internet Engineering Task Force (IETF).
QUIC is a relatively new protocol created to improve performance for connection-oriented web applications beyond what is possible with TCP.
RFC (Request for Comments) was the term for the informal protocol design memos written by the Network Working Group. Although originally expected to be a short-lived set of memos, the RFCs continued and became the formal method of publishing protocol standards and related articles.
SSH (Secure Shell) was designed as a secure replacement for various insecure remote shell protocols; SSH has been a common remote access protocol since 1995.
TCP (Transmission Control Protocol) was the successor to the Network Control Protocol, the original host-host protocol. TCP was byte-oriented, provided two-way, i.e., full-duplex, communication, error checking, and retransmission if bytes were lost in transmission.
TCP Flag Day — Until 31 December 1982, the hosts on the Arpanet all used the Network Control Protocol. Starting on 1 January 1983, the hosts all switched over to Transmission Control Protocol and the underlying Internet Protocol (IP).
TIP (Terminal IMP) was an IMP augmented with 63 ports for terminals. The TIP was the equivalent of an IMP and host computer that implemented the Telnet protocol to reach other hosts on the Arpanet.
UDP (User Datagram Protocol) is a protocol parallel to TCP used when error checking and correction are either not necessary or is performed in the application. Unlike TCP, UDP does not have handshake dialogues. There is no guarantee of delivery, ordering, or duplicate protection.

TABLE 1. Terminology.

The plan evolved over the next few years. The key decisions were to use packet switching and separate small computers, designated interface message processors (IMPs), as routers. Finally, in 1968, a formal Request for Quotations (RFQ) was issued asking companies to design, build, and operate this network. Bolt, Beranek, and Newman (BBN) in Massachusetts was selected and began work in early 1969.

Although there had been a lot of thought regarding the communications subnet, there were

Unlike earlier small-scale network projects, the Arpanet project involved computers from several different vendors. This heterogeneity forced the creation of a common hardware interface.

The Timeline

While this article is primarily focused on the original Arpanet, parts of this report card are best understood in the context of both the Arpanet and the Internet. Here's a very selective and compact timeline.

- 1965-68: Arpanet planning
- August 1968: Request for Quotations for the Interface Message Processors (IMPs) and communications subnet
- January 1969: Bolt, Beranek and Newman begins work on the IMPs and subnet
- September through December 1969: First four nodes installed at UCLA, SRI, UCSB, and Utah
- October 1971: Bake-off at MIT to test interoperability of the implementations of NCP and Telnet
- October 1972: Public display of the Arpanet at the International Conference on Computer Communication (ICCC) meeting in Washington, DC.
- May 1974 first TCP paper
- 1 January 1983: TCP Flag Day

For a more complete timeline, see "A Short History of the Internet," https://www.internetsociety.org/internet/history-internet/

still areas that needed development. Two of the obvious hurdles were the lack of suitable hardware interfaces for communication between computers and the differences in formats and data representation.

There were no standard interfaces and no commonality across the operating systems. Operating systems of the day lived at the centers of their respective worlds. They had not been designed with a means of peering with other computers, so one of our challenges was developing hardware interfaces and the software structure to support peering relationships across these diverse systems.

Unlike earlier small-scale network projects, the Arpanet project involved computers from several different vendors. This heterogeneity forced the creation of a common hardware interface. For the Arpanet, BBN designed a simple bit-serial interface. Each site designed and fabricated its half of the connecting hardware. After the first several of these were built as ad hoc projects within the research laboratories, commercial interfaces became available for the most common machines, notably the Digital Equipment Corporation (DEC) PDP-10.

Shortly after the Arpanet project started, there was active research on technologies for local area networks. The Ethernet design prevailed in the marketplace, and Sun Microsystems unveiled its initial product line with Ethernet interfaces on all its machines. This was a period of vibrant development of local area network technology.

Today, every computer comes with one or more standardized interfaces with the full expectation it will be interacting with other computers, not just peripheral devices. Without question, the goal of getting computers connected to other computers is a complete success and deserves the maximum grade.

Connectivity: A+

Protocols: The software was a different matter. I was part of the team of graduate students from the initial four Arpanet sites tasked with designing the host-level protocols. In a sense, we were the customers or users of the Arpanet, but we were also the builders of the applications that allowed others to use the Arpanet.

We tried to create a set of building blocks that would support multiple applications. We expected others would build on whatever we designed. We expected the protocols would be designed and implemented as a series of layers. We expected most of the layers to be optional, to be used if they were useful for the application and otherwise ignored. These presumptions are at the heart of the open, layered architecture that characterizes the Internet.

We focused on designing a base layer that encompassed a virtual circuit to hide the packetized nature of the underlying communications. We expected the simulation of dial-up to support login and machine-to-machine file transfer to be the applications that would use the base layer. These two protocols, Telnet for remote login and File Transfer Protocol (FTP), did indeed come into existence and have served as primary protocols throughout the history of the Internet. However, in recent years, Secure Shell (SSH) and QUIC have provided substantial improvements in security and performance.

The base protocol was initially called the Host-Host protocol. It later became known as the Network Control Protocol (NCP). Its basic construct was a one-directional ("simplex") bit-serial connection, and a pair of these connections was required for two-way communication. During the period when it was designed, eight-bit bytes had not yet become standard. The NCP was replaced by the Transmission Control Protocol (TCP) as part of the evolution of the Arpanet to a network of networks (i.e., the Internet). By that time, eight-bit bytes had become standard, and bidirectional ("full-duplex") connections in TCP replaced the simplex connections in NCP.

An essential feature of both NCP and TCP was flow control. The IMP subnet had its own flow control to prevent deadlocks due to lack of buffer space, but it was clear early in the design phase the computers at the ends also needed a mechanism to control the consumption of space. We provided controls within the NCP and TCP protocols for the receiving side of a connection to signal how much space was available to the sending side. More on this below.

These protocols were remarkably robust and hence deserved a strong grade. That said, they weren't perfect, and grading them requires some discussion. Here are three parts of the architectural design that didn't work out as expected:

1. The idea of creating a virtual circuit and then simulating a remote terminal connection looked simple at first, but we soon realized the abstraction of a virtual circuit had a weakness. In remote terminal connections, the user must have a way of interrupting the computer's operation. The specific method varies from one operating system to another. Usually, it was a reserved character (e.g., control-C, control-Z, or DEL). For "real" connections, by either hardwire or dial-up circuits, the operating system recognized the signal immediately whenever the user typed an interrupt character. The virtual circuits we defined as the bottom layer of the protocol stack included a variable amount

of buffering. There was no guarantee that a character typed by the user would be seen immediately — or even eventually — by the remote operating system. We therefore had to augment the clean, simple abstraction of virtual circuits by adding a new channel to communicate urgent signals.

2. While we were focused on applications based on virtual circuits, others were focused on real-time communications, particularly packetized speech. For these applications, virtual circuits are a poor match. In the transition from NCP to TCP, a parallel protocol, the User Datagram Protocol (UDP), was designed that did not include flow control and did not attempt to guarantee delivery.
3. In our early thinking about protocols, we imagined processes running on multiple computers cooperating on a common task. We intended the protocol suite to support arbitrary peer-to-peer communications, not just client-server-style applications. The protocols do provide this support, but it has turned out to be used relatively rarely.

Flow control deserves its own discussion, so we close this section on the protocol suite with a decent but not perfect grade.

Protocol Suite: B+

Collaboration: A key part of the vision was collaboration among researchers. The network was envisioned as connecting both computers and people. Licklider and Taylor, the first and third directors of the (D)ARPA Information Processing Techniques Office (IPTO), wrote their 1968 visionary paper, "The Computer as a Communications Device," highlighting the focus on collaboration [5]. Their opening sentence, "In a few years, men [sic] will be able to communicate more effectively through a machine than face to face," stated their vision succinctly.

The DARPA research community enjoyed a fair degree of collegiality. The funding came from a common source, and the Office encouraged collaboration. Although not considered technically demanding, electronic mail quickly became the dominant use of the Arpanet. The prospect of highly interactive shared graphics and shared creation of documents fired the imagination. Douglas Engelbart's landmark work at SRI, the second Arpanet site, combining the invention of the mouse, hypertext, and shared access to documents, was presented at the ACM/IEEE Computer Society Fall Joint Computer Conference in San Francisco in December 1968 [6]. It quickly became known as the "Mother of All Demos" [7]. It would take quite a few years before everyone had the same capabilities, but the direction was set early on.

The tools and techniques for collaboration are still evolving, but it's fair to say results have been spectacular.

Collaboration: A

Extensibility: Collaboration tools are just one example of the open-ended and evolving nature of the network. The direction set within the Arpanet project continued full force into the Internet. Continual change is stressful, though, and doesn't come for free. How well has the protocol suite fared under the stresses of extensibility and scaling? Extensibility has been good, with many new applications and supporting protocols available. User-level data used to be limited to text; it now includes colors, pictures, videos, audio, and more. And extensibility has included adaptation to multiple languages and cultures, localized initiatives, and unlimited new applications. I think it's easy to give a very strong grade.

Extensibility: A

Performance

Functionality is essential but not sufficient. Performance is also essential. The four aspects of performance are interactivity, reliability, scalability, and flow control.

Interactivity: One of the key design parameters for the Arpanet was the requirement to deliver a message within a half-second. ("Message" in this context was the term used for the unit of transfer between a host and an IMP. It was limited to about 8000 bits. Messages were subdivided into packets of approximately 1000 bits or less for transmission over the subnet and reassembled into messages at the destination IMP.) As noted above, the Arpanet connected the several DARPA-supported computer science research labs. Each of these labs had highly interactive time-shared computers. The Arpanet thus extended those environments. It would have been possible to build a network that was much less interactive for the movement of email and files. Indeed, UUNET and BITNET were successful examples of this sort of design [8, 9]. However, the goal for Arpanet included the highly interactive use of remote computers and the eventual inclusion of real-time voice and graphics over the net.

Long-distance, point-to-point leased communication lines were relatively expensive when the Arpanet was designed. This was true even for slow-speed lines operating at 1200 or 2400 bits per second. The Arpanet was designed and built using 50,000 bits per second leased lines, benefiting in part from a reduced tariff available to the government. As a result, in addition to less time-sensitive applications such as email and file transfer, the Arpanet supported the interactive use of remote computers and early experiments in packetized voice communication and interactive graphics.

Maintaining a high degree of interactivity involved numerous technical challenges, including the flow control problems described below. But it also greatly increased the utility of the network. A further benefit is that whenever there were problems in the operation of the network, they tended to show up within seconds, not hours or days.

Interactivity was one of the hallmarks of the Arpanet, an unqualified success.

Interactivity: A

Reliability: A separate concern in the design of the Arpanet was reliability. While most projects funded by the IPTO were intended to create new capabilities and demonstrate what might be possible, some of the projects were intended to serve the dual purpose of advancing the science and to be usable as tools for the research community. Several time-sharing systems were developed within the IPTO community with this dual

Collaboration tools are just one example of the open-ended and evolving nature of the network. The direction set within the Arpanet project continued full force into the Internet. Continual change is stressful, though, and doesn't come for free.

purpose, and the Arpanet project followed the same path.

Inherent in the dual purpose is a certain tension. Researchers interested in how well the network worked and particularly how well it functioned when pushed to its limits wanted to be able to take the network down to test it. However, the rest of the research community wanted the network to be usable whenever they wanted to use it. It was clear from the beginning and became even clearer once the network was functioning that availability was paramount. Periods of deliberate unavailability were scheduled carefully and infrequently. For most in the research community it became a utility. It had to work almost all the time for almost everybody. In a 2019 virtual round table, The Arpanet and Its Impact on the State of Networking, Ben Barker recounts both the necessity and success of improving the reliability of the IMPs from 98 to 99.98 percent [10]. This meant a mere 2 percent improvement in uptime but a 99 percent reduction in downtime. Compared to the 99.999 percent target uptime of commercial telephone systems [11] 99.98 percent may seem modest, but it was sufficient to shift the perception of the Arpanet users from "I can't depend on it" to "It's almost always there when I need it."

The fact that routes were automatically adjusted whenever a link or a router failed resulted in a network that was fairly robust. There were anecdotes of the network continuing to work even when there was a major natural disaster. After the first few years, widespread outages were rare to non-existent. The Arpanet, as a utility, was far more robust than the computers connected to it. The early robustness of the Arpanet resulted in both an expectation and the experience base to continue a high standard of robustness even through the transition from the Arpanet to the Internet and the dramatic scaling that followed.

Reliability: A–

Scalability: Scalability, on the other hand, has been a more complex story. On one hand, the Internet now connects several billion users, roughly half the entire human population [12]. The growth has been larger and faster than originally anticipated. The address space for designating hosts was expanded from eight bits (i.e., a maximum of 256 hosts) to 32 bits in IPv4 (i.e., a maximum of four billion hosts), and then again to 128 bits — a very large number — in IPv6. Unfortunately, this latter transition has not been smooth, and the Internet today operates with an awkward mixture of IPv4 and IPv6 transport protocols. With sadness, I must assign just a middling grade.

Scalability: B–

Flow Control: Finally, we come to a difficult part of the story, flow control. As mentioned above, we discovered right away that even though the IMPs implemented flow control to protect the subnet from congestion, similar controls were needed to manage the flow between the hosts. When we designed the flow control within the NCP, we were conscious of the wide range of capabilities across the collection of hosts. Very small hosts, with the TIP as the premier example, had very little buffer space but were able to respond to interrupts without much delay. Very large hosts, with Multics as the premier example, had far more capacity but treated interrupts as heavy-duty context switches.

We weren't sure whether to use bits or messages as the unit of control between hosts, so we punted and provided both. That is, the receiving side of a connection sent the sending host separate allocations of bits and messages; the sender would then keep track of how many bits and how many messages it had sent. If either quantity was exhausted, the sender would pause and await a fresh allocation.

We didn't know what the right settings would be. We simply provided the mechanism and hoped there would be either practical experience or insightful analysis by engineers schooled in control theory or other relevant disciplines.

In the transition from NCP to TCP, the metering of bits changed to the metering of bytes, and the metering of messages was dropped. That was a simplification but not a solution. As the network grew and traffic loads increased, there were an increasing number of cases where the performance was poor. Throughput was observed to be just a minuscule fraction of the channel capacity.

Van Jacobsen and his colleagues analyzed the flow control in TCP and identified multiple reasons for congestion and delay [13]. They developed some principles and algorithms. Performance improved dramatically. But that, unfortunately, is not the end of the story.

An ideal scenario for the continuous flow of data is having enough space allocated at each of the hops along the path so packets can move forward at the same rate they enter the system. Feedback as the connection is established is used to set the flow control parameters.

This ideal scenario suffers from two sources of interference. First, there are multiple layers of protocols, and from a performance perspective they interact. A TCP connection may travel part of the way over Wi-Fi, part of the way over a wide area network, and part of the way through an enterprise network. Each of these has its own buffering and flow control strategies. We have neither adequate theory nor fully practical tools to know how to set the parameters at each level of the protocol stack.

The second source of interference is an embarrassment of riches. In sharp contrast to the early days of the network when memory space in computers was always very tight, memory space has become plentiful. It has become commonplace in many parts of the network to allocate more buffer space than is needed for ideal flows. Consequently, when there is congestion, it isn't detected right away. Queues build up and take a long time to drain. The colorful term "bufferbloat" describes this phenomenon [14].

Protocol designers, router vendors, end system implementors, and network operators continue to wrestle with bufferbloat. We need a more complete theory and improvements in the control structures within the protocols.

With respect to the work done within the Arpanet project, this aspect was addressed and thus deserves a minimally passing grade, but the results were not yet sufficient. Charitably, we can say the Arpanet experience demonstrated how complicated and challenging this topic is.

Flow Control: C–

Evolution

This last section grades the social processes related to design, testing, and standards setting. As noted above, the Arpanet was funded and managed as a research project to develop both the technology and to serve as a usable service to the IPTO research community. During the 1965–1968 planning period, the focus of attention was on the architecture of the packet-switching subnet, including the use of separate computers to serve as routers. The creation of host-level protocols and the software to implement them was left as a task for the computer scientists at the Arpanet sites to address. Questions of how these protocols would be managed, whether there would be rules governing changes, and how the implementations would be tested were not planned out in advance. Instead, IPTO management left room for the research community to address these issues. At the same time, IPTO management retained the power of the purse. When successful initiatives emerged within the community, IPTO was able to provide whatever funding was needed to keep those initiatives going. Equally, had the efforts within the research community not succeeded on their own, IPTO retained the option of recruiting and funding experts to help out.

Three distinct organizational or "social" processes emerged:

1. Design by distributed ad hoc groups
2. Testing by interoperability instead of compliance
3. Standards based on market acceptance instead of law

Design: The protocol stack was developed at first by the graduate students associated with the first Arpanet sites. There was no formal structure. A handful of us met, engaged in broad discussion at first, and gradually focused on the specific details of the NCP, Telnet, and FTP. Our initial set of notes were called Requests for Comments (RFCs), partly out of recognition that we didn't have any formal authority [15]. All of the interactions were open. Anyone was welcome to attend, anyone was welcome to contribute, and the documents were available to anyone.

In the early days, there weren't any commercial vendors designing or building network products. Creation of a new protocol was initiated by one or more people interested in creating a new capability. The IPTO research community used a wide variety of computers, so any protocol that was going to be useful across the community would require multiple implementations. This created an environment where it was advantageous for multiple people to participate in the design and for the design to be as simple as possible.

The layered, open architecture has made it possible for independent groups to create protocols without advance permission and with almost no coordination. Experimental protocols are created frequently. Some evolve into mainstream standards; others remain experimental or are used by a limited group. The only coordination that's required is in the assignment of identifiers such as a protocol number or a port assignment. A very lightweight administrative system came into existence to administer these assignments, now part of the standards process described below.

The openness of the design process has made possible the explosion of network applications and is the embodiment of permissionless innovation. The openness of the design process is arguably even more important than any of the specific elements of the architecture and design, and deserves an extraordinary grade.

Design Process: A+

Testing: As described above, it was necessary to implement each protocol on several different computers. At first, we didn't think through how to get the implementations to work with each other. In October 1971, the Massachusetts Institute of Technology (MIT) hosted a "bake-off" to test the NCP and Telnet implementations. Each site sent a representative to MIT and attempted to connect their host to each other host. At the end of two days of testing, almost all the hosts were able to connect to each other.

What we didn't have at the bake-off was a reference implementation. Each implementation was tested against the others, not just against a designated "gold standard." This approach avoided the creation of a compliance testing regime that usually imposes costs and delays. Later, as the Arpanet transformed into the Internet, interoperability testing was embodied in the Interop conferences starting in 1986 [16].

The emergence of interoperability testing, as opposed to compliance, is one of the less heralded but crucial factors that contributed to the Arpanet's and Internet's success.

Interoperability Testing: A

Standards: The Arpanet was a research project funded at first entirely under the aegis of DARPA's IPTO. As noted above, the users of the network were also the developers of the initial suite of host-level protocols (e.g., NCP, Telnet, NCP). Consensus among the developers was reached informally, with IPTO watching at a distance to see if the work was progressing. Documentation in the form of RFCs was shared quickly and openly. This ad hoc arrangement served as a de facto standards process. A bit later, during the mid-1970s, the traditional standards organizations, CCITT (now the International Telecommunication Union, ITU), IEEE, and International Standards Organization (ISO), became involved in the standardization of subsequent protocols such as X.25, Ethernet, and Open Systems Interconnection (OSI).

However, the original informal process that was central to the protocol developments for the Arpanet continued. The Network Working Group, which had started with around a dozen people, continued to grow and evolve, formalized in 1986 as the Internet Engineering Task Force (IETF). Meetings grew to between 1000 and 2000 people attending three times per year. RFCs became the accepted forum for publishing protocol specifications. A lightweight process was created to determine when a protocol specification had reached maturity and gained consensus to be designated as an Internet Standard [17].

The IETF has taken on some but not all of the formal attributes of the pre-existing standards organizations. Participation in the IETF processes is through individuals, not companies or govern-

ments, and adoption of its standards is driven by market forces as opposed to government requirements.

Of particular note is the openness of the IETF process:

1. The architecture is open. Additions are welcomed. There are roughly 100 active working groups at any given time.
2. Participation is open. Everyone is welcome to join any working group or start a new working group. New protocol designs are judged for completeness and safety, but otherwise, there aren't any barriers to creating new protocols.
3. The documents are open. All documentation, including working documents, is available to anyone anywhere without cost.

Not included in the creation of the standards process was any form of enforcement. Instead of regulation, market forces determine the success or failure of new protocols.

Standards Process: A

Conclusion

Without question, the Arpanet was a huge success. It provided revolutionary capabilities in its own right, and it opened the door for the creation and explosive growth of the Internet. And like any large project, it spawned many new projects, with more report cards to be written. Over the next 50 years, the functionality and utility of the Internet will almost certainly continue to grow. It will have challenges along the way — technical challenges such as flow control as well political challenges as more entities try to assert control over the network. However, with this report card, it is clearly ready to graduate to the next level.

Acknowledgments

I'm grateful for the comments and collaborative assistance of friends and colleagues: Scott Bradner, Vint Cerf, Dave Crocker, Heather Flanagan, Robert Kahn, Dan Lynch, Bob Metcalfe, Dave Reed, and Dave Täht. Errors, omissions, confusions, and such are solely mine.

References

[1] M. Mitchell Waldrop, *The Dream Machine*, Stripe Press, 2001.
[2] D. Hemmendinger, "LISP," *Encyclopedia Britannica*, n.d.; https://www.britannica.com/technology/LISP-computer-language, accessed 20 July 2021.
[3] Britannica, T. Editors of Encyclopaedia, "BASIC," Encyclopedia Britannica, n.d.; https://www.britannica.com/technology/BASIC, accessed 20 July 2021.
[4] K. Hafner and M. Lyon, *Where Wizards Stay up Late*, Simon & Schuster Paperbacks, 2006.
[5] J. Licklider and R. Taylor, "The Computer as a Communications Device," *Science and Technology*, vol. 76, 1968, pp. 21–31.
[6] D. Engelbart and W. English, "A Research Center for Augmenting Human Intellect," *Int'l. Wksp. Managing Requirements Knowledge,* , San Francisco, CA, 1968, p. 395; https://dx.doi.org/10.1109/AFIPS.1968.52.
[7] E. S. Hintz, "The Mother of All Demos," Smithsonian Institution,10 Dec. 2018; https://invention.si.edu/mother-all-demos, accessed 20 July 2021.
[8] Wikipedia Contributors, "UUNET," *Wikipedia, The Free Encyclopedia*; https://en.wikipedia.org/w/index.php?title=UUNET&oldid=1013880594, accessed July 20, 2021.
[9] Wikipedia Contributors, "BITNET," *Wikipedia, The Free Encyclopedia*; https://en.wikipedia.org/w/index.php?title=BITNET&oldid=1031287836, accessed July 20, 2021.
[10] S. D. Crocker, "The Arpanet and Its Impact on the State of Networking," *IEEE Computer*, vol. 52, no. 10, Oct. 2019, pp. 14–23; http://dx.doi.org/10.1109/MC.2019.2931601.
[11] Wikipedia Contributors, "Plain Old Telephone Service," *Wikipedia, The Free Encyclopedia*; https://en.wikipedia.org/w/index.php?title=Plain_old_telephone_service&oldid=1047467774, accessed Oct. 2, 2021.
[12] J. Johnson, "Global Digital Population as of January 2021 (in Billions)," *Statista*, 7 Apr. 2021; https://www.statista.com/statistics/617136/digital-population-worldwide/, accessed 19 July 2021.
[13] V. Jacobson, "Congestion Avoidance and Control," *Symp. Proc. Commun. Architectures and Protocols — SIGCOMM '88*, 1988; http://dx.doi.org/10.1145/52324.52356.
[14] The Bufferbloat Community, "Bufferbloat"; https://www.bufferbloat.net/projects/, accessed 20 July 2021.
[15] RFC Editor, "History"; https://www.rfc-editor.org/history/, accessed 19 July 2021.
[16] J., Markoff, "Up from the Computer Underground," The New York Times, 27 Aug. 1993, https://www.nytimes.com/1993/08/27/business/up-from-the-computer-underground.html, accessed 29 Sept. 2021.
[17] "Standards Process," IETF; https://www.ietf.org/standards/process/, accessed 27 Oct. 2021.

Biography

Steve Crocker (steve@shinkuro.com) is an Internet pioneer who helped develop the protocols for the Arpanet and created the RFC Series that document Internet protocols. He has worked in research, government, and business for the past 50 years. He has specialized in network security research. He was the founding Chair of ICANN's Security and Stability Committee, and was a member of the ICANN Board and Chair of the Board from 2011 to 2017. He holds a B.A. in mathematics and a Ph.D. in computer science from UCLA. For the past four years, he has been working with a small group to develop the framework for expressing and analyzing registration data directory service policies.